

II – Distributed Cryptography

David Pointcheval

Ecole normale supérieure, CNRS & INRIA



ENS – Paris – 2018

- 1 Secret Sharing**
 - Introduction
 - Shamir Secret Sharing
 - Verifiable Secret Sharing

- 2 Distributed Cryptography**
 - Introduction
 - Distributed Decryption
 - Distributed Signature
 - Distributed Key Generation

Outline

- 1 Secret Sharing**
 - Introduction
 - Shamir Secret Sharing
 - Verifiable Secret Sharing

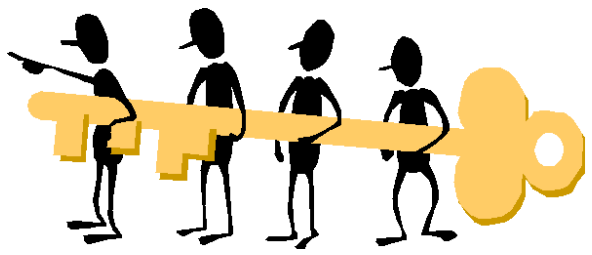
- 2 Distributed Cryptography**

Key Management

In case of a critical private key (decryption or signing key)

- **Abuse**: one user can use the secret key alone
- **Loss**: in case of loss of the key (destruction)

⇒ share the secret key between several users



Let $S \in \{0, 1\}^\ell$ be a secret bit-string to be shared between two people (Alice and Bob):

- one chooses a random $S_1 \in \{0, 1\}^\ell$, and sends it to Alice
- one computes $S_2 = S \oplus S_1$, and sends it to Bob

Security:

- Alice knows a random value
- Bob knows a value masked by a random value: a random value!

⇒ individually, they have no information on S

Together, they can recover $S = S_1 \oplus S_2$

Let $S \in \{0, 1\}^\ell$ be a secret bit-string to be shared between n people (U_1, \dots, U_n):

- one chooses random values $S_i \in \{0, 1\}^\ell$, for $i = 1, \dots, n - 1$ and sends S_i to U_i
- one computes $S_n = S \oplus S_1 \oplus \dots \oplus S_{n-1}$, and sends it to U_n

Security:

- U_1, \dots, U_{n-1} know random values
- U_n knows a value masked by random values: a random value!

⇒ individually, they have no information on S

⇒ but also, any subgroup of $(n - 1)$ people has no information on S

All together, they can recover $S = S_1 \oplus \dots \oplus S_n$

Unconditional Security

Outline

Any subgroup of $(n - 1)$ people has no information on S !
⇒ if one people does not want / is not able to cooperate:

S is lost forever!

Threshold Secret Sharing

(n, k) -Threshold Secret Sharing

A secret S is shared among n users:

- any subgroup of k people (or more) can recover S
- any subgroup of less than k people has no information about S

1 Secret Sharing

- Introduction
- Shamir Secret Sharing
- Verifiable Secret Sharing

2 Distributed Cryptography

Lagrange Interpolation of Polynomials

Let us be given k points (x1, y1), ..., (xk, yk), with distinct abscissa. There exists a unique polynomial P

- of degree k – 1
- such that P(xi) = yi for i = 1, ..., k

$$L_j(X) = \prod_{\substack{i=1 \\ i \neq j}}^{i=k} \frac{X - x_i}{x_j - x_i} \quad \begin{cases} L_j(x_j) = 1 \\ L_j(x_i) = 0 \text{ for all } i \neq j \end{cases}$$

As a consequence:

$$P(X) = \sum_{j=1}^k y_j L_j(X) \text{ satisfies } \begin{cases} \text{deg}(P) = k - 1 \\ P(x_i) = y_i \quad \forall i = 1, \dots, k \end{cases}$$

For any subset S of k indices:

$$L_{S,j}(X) = \prod_{\substack{i \in S \\ i \neq j}} \frac{X - x_i}{x_j - x_i} \quad \begin{cases} L_{S,j}(x_j) = 1 \\ L_{S,j}(x_i) = 0 \text{ for all } i \in S, i \neq j \end{cases}$$

and

$$P(X) = \sum_{j \in S} y_j L_{S,j}(X) : S = P(0) = \sum_{j \in S} y_j L_{S,j}(0)$$

If one notes λS,j = LS,j(0) (that can be publicly computed)

$$x = \sum_{j \in S} y_j \lambda_{S,j}$$

Outline

- 1 Secret Sharing
 - Introduction
 - Shamir Secret Sharing
 - Verifiable Secret Sharing

2 Distributed Cryptography

Verifiable Secret Sharing

[Chor-Goldwasser-Micali-Awerbuch – FOCS '85]

If Eve claims she shared her decryption key: how can we trust her?

- we try to recover the key?
- how to do without revealing additional information?

⇒ Verifiable Secret Sharing

For DL Keys [Feldman – FOCS '87]

Eve's keys are, in a group G = <g> of prime order q,
sk = x pk = y = g^x

(n, k)-Secret sharing: x = P(0) for P(X) = sum_{i=0}^{k-1} ai Xi
=> Si = P(i) for i = 1 ... , n

For any subset S of k indices:

- x = sum_{j in S} Sj λS,j
- y = g^x = g^{sum_{j in S} Sj λS,j} = prod_{j in S} (g^{Sj})^{λS,j} = prod_{j in S} v_j^{λS,j} for v_j = g^{Sj}

For DL Keys [Feldman – FOCS '87]

Eve's keys are, in a group $\mathbb{G} = \langle g \rangle$ of prime order q ,

$$sk = x \quad pk = y = g^x$$

(n, k) -Secret sharing: $x = P(0)$ for $P(X) = \sum_{i=0}^{k-1} a_i X^i$

- Eve computes $S_i = P(i)$ for $i = 1 \dots, n$ and $v_i = g^{S_i}$
- Eve sends each S_i privately to each U_i
- Eve publishes $v_i = g^{S_i}$ for $i = 1, \dots, n$
- Each U_i can then check its own v_i w.r.t. to its S_i
- Anybody can check
$$y = \prod_{j \in \mathcal{S}} v_j^{\lambda_{\mathcal{S},j}}$$

for any subset \mathcal{S} of size k

- 1 Secret Sharing
- 2 Distributed Cryptography
 - Introduction
 - Distributed Decryption
 - Distributed Signature
 - Distributed Key Generation

Secret Sharing vs. Distributed Cryptography

Outline

If Eve shares her decryption key sk ,
 the (U_i) will have to cooperate to recover the key sk
 and then decrypt the ciphertext

But then, they all know the decryption key sk !

How can the (U_i) use their shares (S_i) to decrypt (or sign),
 without leaking any additional information about sk ?

⇒ Multi-party computation

Let us try on ElGamal decryption (with shared DL keys)

- 1 Secret Sharing
- 2 Distributed Cryptography
 - Introduction
 - Distributed Decryption
 - Distributed Signature
 - Distributed Key Generation

ElGamal Encryption

In a group $\mathbb{G} = \langle g \rangle$ of order q

- $\mathcal{K}(\mathbb{G}, g, q)$: $x \xleftarrow{R} \mathbb{Z}_q$, and $sk \leftarrow x$ and $pk \leftarrow y = g^x$
- $\mathcal{E}_{pk}(m)$: $r \xleftarrow{R} \mathbb{Z}_q$, $c_1 \leftarrow g^r$ and $c_2 \leftarrow y^r \times m$.
Then, the ciphertext is $c = (c_1, c_2)$
- $\mathcal{D}_{sk}(c)$ outputs c_2/c_1^x

We assume an (n, k) -secret sharing of x
and a qualified set \mathcal{S} : $x = \sum_{j \in \mathcal{S}} S_j \lambda_{S,j}$
 $\mathcal{D}_{sk}(c) = c_2/c_1^x$: one needs to compute c_1^x

$$c_1^x = c_1^{\sum_{j \in \mathcal{S}} S_j \lambda_{S,j}} = \prod_{j \in \mathcal{S}} (c_1^{S_j})^{\lambda_{S,j}}$$

Each user computes $C_j = c_1^{S_j}$, and then $c_1^x = \prod_{j \in \mathcal{S}} C_j^{\lambda_{S,j}}$

In a group $\mathbb{G} = \langle g \rangle$ of order q

- $\mathcal{K}(\mathbb{G}, g, q)$: $x \xleftarrow{R} \mathbb{Z}_q$, and $sk \leftarrow x$ and $pk \leftarrow y = g^x$
- $\mathcal{E}_{pk}(m)$: $r \xleftarrow{R} \mathbb{Z}_q$, $c_1 \leftarrow g^r$ and $c_2 \leftarrow y^r \times m$.
Then, the ciphertext is $c = (c_1, c_2)$
- $\mathcal{D}_{sk}(c)$ outputs c_2/c_1^x

Given a qualified set \mathcal{S} : $x = \sum_{j \in \mathcal{S}} S_j \lambda_{S,j}$
Each user computes $C_j = c_1^{S_j}$, and then $c_1^x = \prod_{j \in \mathcal{S}} C_j^{\lambda_{S,j}}$

Assume Charlie a.k.a. U_1 , sends a random C_1 :

- the others will compute a wrong decryption
- Charlie will be able to extract the plaintext!

Fraud Detection

Each user computes $C_j = c_1^{S_j}$, and then $c_1^x = \prod_{j \in \mathcal{S}} C_j^{\lambda_{S,j}}$

But U_1 , sends a random C_1 : instead of $c_1^{S_1}$, knowing also $v_1 = g^{S_1}$
 \implies Decide a DDH tuple (g, c_1, v_1, C_1)

Robustness

A defrauder can be detected

\implies Proof of DDH membership for the tuple (g, c_1, v_1, C_1) ,
without leakage of any information about S_1

NIZK Diffie-Hellman Language

In a group $\mathbb{G} = \langle g \rangle$ of prime order q ,
the **DDH** (g, h) assumption states it is hard to distinguish
 $\mathcal{L} = (u = g^x, v = h^x)$ from $\mathbb{G}^2 = (u = g^x, v = h^y)$

- \mathcal{P} knows x , such that $(u = g^x, v = h^x)$ and wants to prove it
- \mathcal{P} chooses $k \xleftarrow{R} \mathbb{Z}_q^*$, sets $U = g^k$ and $V = h^k$
- \mathcal{P} computes $h = \mathcal{H}(g, h, u, v, U, V) \in \mathbb{Z}_q$
- \mathcal{P} computes $s = k + xh \text{ mod } q$

The proof consists of the pair (h, s) :
anybody can check whether $h = \mathcal{H}(g, h, u, v, g^s u^h, h^s v^h)$

This proof allows to detect the defrauder

1 Secret Sharing

2 Distributed Cryptography

- Introduction
- Distributed Decryption
- Distributed Signature
- Distributed Key Generation

Schnorr Signature

- $\mathbb{G} = \langle g \rangle$ of order q and $\mathcal{H}: \{0, 1\}^* \rightarrow \mathbb{Z}_q$
- Key Generation $\rightarrow (y, x): x \in \mathbb{Z}_q^*$ and $y = g^{-x}$
- Signature of $m \rightarrow (r, h, s)$
 $k \xleftarrow{R} \mathbb{Z}_q^* \quad r = g^k \quad h = \mathcal{H}(m, r) \quad s = k + xh \text{ mod } q$
- Verification of (m, r, s)
 compute $h = \mathcal{H}(m, r)$ and check $r \stackrel{?}{=} g^s y^h$

We assume an (n, k) -secret sharing of x (with the v_i)
 and a qualified set $\mathcal{S}: x = \sum_{j \in \mathcal{S}} S_j \lambda_{\mathcal{S}, j}$
 The users generate a common r and then sign (m, r)
 with a partial signature s_i under v_i :
 \implies the linearity leads to a global signature

Distributed Schnorr Signature

- $\mathbb{G} = \langle g \rangle$ of order q and $\mathcal{H}: \{0, 1\}^* \rightarrow \mathbb{Z}_q$
- Key Generation $\rightarrow (y, x): x \in \mathbb{Z}_q^*$ and $y = g^{-x}$
 We assume an (n, k) -secret sharing of x (with the $v_i = g^{S_i}$)
 and a qualified set $\mathcal{S}: x = \sum_{j \in \mathcal{S}} S_j \lambda_{\mathcal{S}, j}$
- Signature of $m \rightarrow (r, h, s)$
 - each U_i chooses $k_i \xleftarrow{R} \mathbb{Z}_q^*$ and publishes $r_i = g^{k_i}$
 - they all compute $r = \prod r_i \lambda_{\mathcal{S}, j}$ and $h = \mathcal{H}(m, r)$
 - each U_i computes and publishes $s_i = k_i + S_i h \text{ mod } q$
 Then, $s = \sum s_i \lambda_{\mathcal{S}, i}$
- Verification of (m, r, s)
 compute $h = \mathcal{H}(m, r)$ and check $r \stackrel{?}{=} g^s y^h$

Each partial signature (m, r_i, s_i) can be checked: $r_i \stackrel{?}{=} g^{s_i} v_i^h$

Outline

1 Secret Sharing

2 Distributed Cryptography

- Introduction
- Distributed Decryption
- Distributed Signature
- Distributed Key Generation

In the previous schemes (ElGamal encryption and Schnorr signature) the keys are generated in a centralized way:

someone knows the secret key!

Distributed cryptography should include a distributed key generation: the secret key should never exist in one place.

(n, n) -Threshold DL Key Generation

- $\mathbb{G} = \langle g \rangle$ of order q
 - Key Generation $\rightarrow (y, x)$:
 - each U_i chooses $x_i \xleftarrow{R} \mathbb{Z}_q^*$ and publishes $y_i = g^{x_i}$
 - anybody can compute $y = \prod y_i = g^{\sum x_i}$
- The public key y corresponds to the “virtual” secret key
- $$x = \sum x_i \pmod q$$

(n, k) -Threshold DL Key Generation

- $\mathbb{G} = \langle g \rangle$ of order q
- Key Generation $\rightarrow (y, x)$:
 - each U_i chooses a polynomial P_i of degree $k - 1$, and sends $S_{i,j} = P_i(j)$ to U_j
 - each U_j can then compute $S_j = \sum_i S_{i,j} = \sum_i P_i(j) = P(j)$, where $P = \sum_i P_i$
 - each U_j computes and publishes $v_j = g^{S_j}$

The $(S_j)_j$ are an (n, k) -secret sharing of the “virtual” secret key x , corresponding to the public key y , that anybody can compute: For any qualified set \mathcal{S} :

- Secretly: $x = \sum_{j \in \mathcal{S}} S_j \lambda_{\mathcal{S},j} \pmod q$
- Publicly: $y = \prod_{j \in \mathcal{S}} v_j^{\lambda_{\mathcal{S},j}}$